Customer Churn Reduction

Somnath Mahato

August 22, 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

The Customers Churn prediction is an effective measure and research topic for the Telecom Industry as retaining the existing customers is easier then acquiring new ones. The acquisition of new customers involve considerable amount of resources, while retaining existing customers is cost effective and an optimized option for the industries to look upon parameters that can favour both the sides and create improvents in the customers satisfication. This project aims to look into the parameters that can effect the Churning out of customers with Machine Learning algorithms and a detailed analysis on the various important parameters that involves the Customers Churning and their behaviour.

1.2 Data

Data is described upon parameters such as the geographical location, various charges involved, plans provided and the number of customer service calls that decide upon the Churning. The table represents a sample of various fields available in the data.

Table 1.1: Customer Churn Data (Columns 1- 5)

State	Account Length	Area Code	Phone Number	International Plan
HI	101	510	354-8815	no
MT	137	510	381-7211	no
OH	103	408	411-9841	no

Table 1.2: Customer Churn Data (Columns 6- 10)

Voice Mail Plan	Number Vmail Messages	Total Day Minutes	Total Day Calls
no	0	70.9	123
no	0	223.6	86
yes	29	294.7	95

Table 1.3: Customer Churn Data (Columns 11- 14)

Total Night Minutes	Total Night Calls	Total Night Charge	Total Intl Minutes
236	73	73	10.62
94.2	81	139	4.24
300.3	127	105	13.51

Table 1.4: Customer Churn Data (Columns 15- 19)

Total Intl Calls	Total Intl Charge	Number Customer Service Calls	Churn
3	2.86	3	False.
7	2.57	0	False.
6	3.7	1	False.

The table represent the features used in the training and analysis

Table 1.5: Predictor Variables

No.	Features
1	state
2	account length
3	area code
4	international plan
5	voice mail plan
6	total day minutes
7	total day calls
8	total eve calls
9	total night minutes
10	total intl minutes
11	total intl calls
12	number customer service calls

1.2.1 Visualizations

We tried to plot the number of customers by the state. The heatmap Fig:1.2 represents the number of customers per state Python code :B.6
There are numerous customer in the North Eastern State of West Virgina

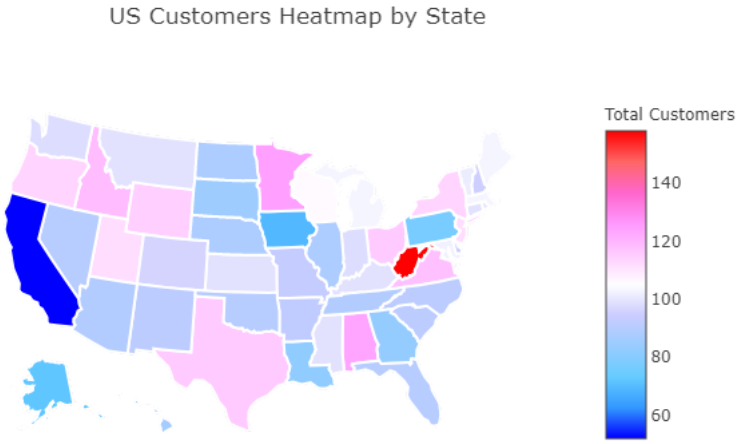

Figure 1.1: Customers Heatmap

and the least customers index being California.

1.2.2 Preprocessing

All Machine Learning Algorithms take the data in a Guassian/Normal distribution for better prediction, accuracy and less variance among features. Eventhough, the most of the features are normally distributed we have scaled all the features for feeding data into model. The scaled data can be as shown

```
X.head()
```

state	account length	area code	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls
-0.675476	0.698941	-0.519166	-0.323240	1.667120	1.273145	1.573802	0.502824	1.574074	-0.064032	-0.060077	-0.063849	0.876999	-0.446928
0.608134	0.169849	-0.519166	-0.323240	1.667120	1.346973	-0.346802	1.158422	-0.347082	-0.101621	0.141693	-0.101089	1.068992	0.154374
0.337900	0.925695	-0.519166	-0.323240	-0.599837	-0.572549	1.171125	0.704546	1.171286	-1.571562	0.494791	-1.572084	-0.748012	0.204483
0.608134	-0.409634	-0.685024	3.093675	-0.599837	-0.572549	2.210292	-1.463971	2.210457	-2.744745	-0.614946	-2.745155	-0.069110	-0.547145
0.675692	-0.636388	-0.519166	3.093675	-0.599837	-0.572549	-0.252163	0.654116	-0.252115	-1.035419	1.100103	-1.034426	-0.267041	1.056327

Figure 1.2: Scaled data after preprocessing

Chapter 2

Methodology

2.1 Preprocessing

Data Exploration is an essential part of every Data Science project. The obtained data may not be always in the required form to perform any analysis or get insights. Most of the machine learining algorithms requires the data to be cleaned, processed or scaled to train a model on it. These techniques are easy to understand but every dataset is different and has a unique challenge in it.

2.1.1 Outlier Analysis

The shown boxplot Fig: 2.1 refers outliers on the predictors variables, we can see various outliers associated with the features. Eventhough, the data has considerale amount of outliers, the approach is to retain every outlier and grab respective behaviour of all customers. As shown there are significant amount of outliers present in the amount of night calls, which indicates a trend on customers' behaviour, there can be normal customers with an average usage appearing within the inter quartile, as well as customers who have business type accounts may have heavy usage and they appear above the quartile which seems important and can be concluded that Outliers here have information and retaining them would have advantage over the analysis.

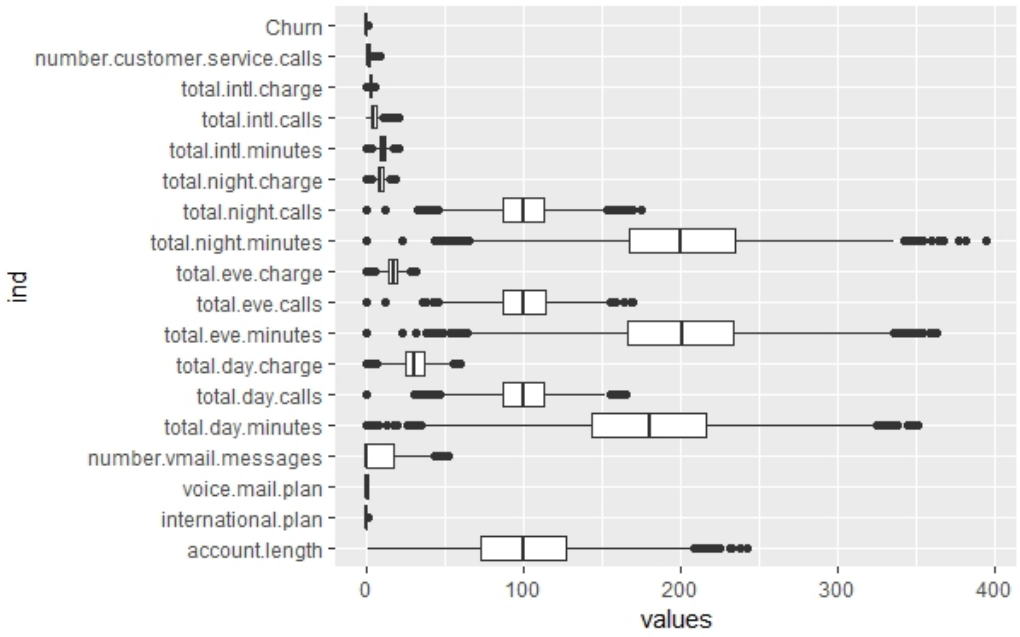

Figure 2.1: Boxplot on Predictors

2.1.2 Feature Engineering

Feature Engineering is described as the knowledge extraction process, where important features are selected using domain knowledge to make a machine learning algorithm work. There can be features that aren't relevant for the analysis, we can remove such variables using numerous ways. However, we considered taking correlation on the variables and make a heatmap Fig: 2.2 to check relationships among the features and then dropping redundant variables

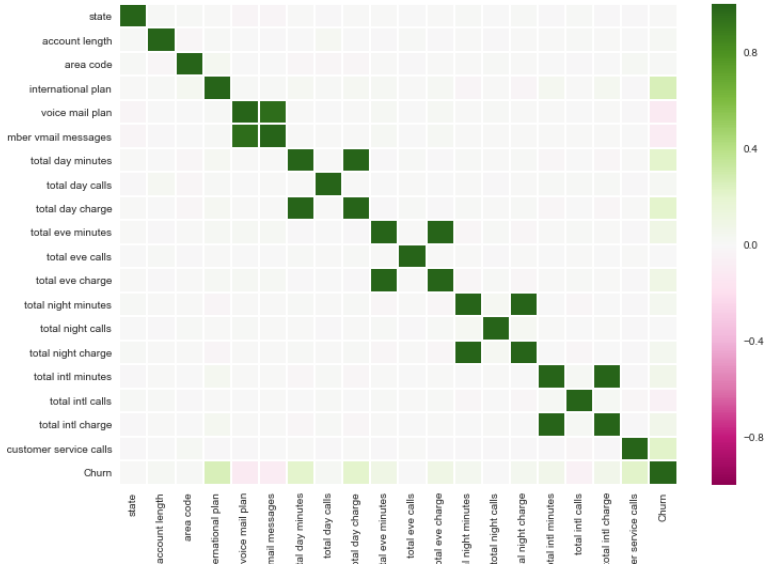

Figure 2.2: Heatmap on Predictors

2.1.3 Feature Selection

The Bi-Variate analysis in Fig: 2.3 shows International Plan, Total Day Minutes, Number of Customer Service Call are related with Churn which is shown in the Linear Plot

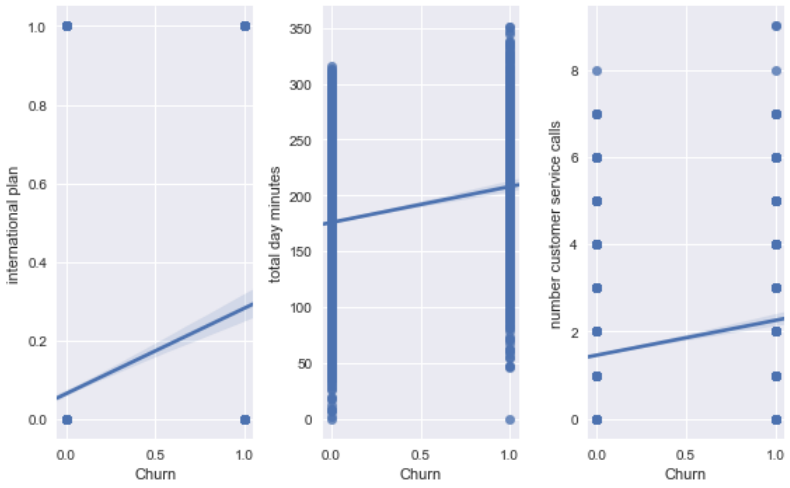

Figure 2.3: Univariate Analysis for Target Variables

The variables Number of Vmail Messages, Total Day Charge,Total Eve Charge,Total Night Charge,Total Intl Charge are dropped from the analysis as they are highly correlated with other variables. The final variables selected for the analysis are Fig: 2.4.

```
customer_df.head()
```

	state	account length	area code	international plan	voice mail plan	total day minutes	total day calls	total eve minutes	total eve calls	total night minutes	total night calls	total intl minutes	total intl calls	number customer service calls	Churn
0	16	128	415	0	1	265.1	110	197.4	99	244.7	91	10.0	3	1	0
1	35	107	415	0	1	161.6	123	195.5	103	254.4	103	13.7	3	1	0
2	31	137	415	0	0	243.4	114	121.2	110	162.6	104	12.2	5	0	0
3	35	84	408	1	0	299.4	71	61.9	88	196.9	89	6.6	7	2	0
4	36	75	415	1	0	166.7	113	148.3	122	186.9	121	10.1	3	3	0

Figure 2.4: Selected Features for analysis

Chapter 3

Modeling

3.1 Ensemble Technique

The goal of any machine learning problem is to find a single model that will best predict our wanted outcome. Rather than making one model and hoping this model is the best/most accurate predictor we can make, ensemble methods take a myriad of models into account, and average those models to select one final model. Our approach is to consider the classification models such as Logistic Regression, K-Nearest Neighbors, Decision Tree and The Random Forest model to obtain all the accuracies and then select a model that has a better Accuracy, Precision and Recall for a binomial prediction of whether or not a Customer will Churn. We used an iterative approach to test upon the above models and then plotted the results with accuracy,precision and recall respectively. The results from the model can be as shown, Python

Decision_Trees

	precision	recall	f1-score	support
0	0.96	0.95	0.95	1297
1	0.68	0.74	0.71	203
avg / total	0.92	0.92	0.92	1500

KNN

	precision	recall	f1-score	support
0	0.90	0.98	0.94	1297
1	0.67	0.27	0.39	203
avg / total	0.87	0.88	0.86	1500

Logistic

	precision	recall	f1-score	support
0	0.88	0.97	0.92	1297
1	0.45	0.18	0.26	203
avg / total	0.82	0.86	0.83	1500

Figure 3.1: Model Results 3/4

RF

	precision	recall	f1-score	support
0	0.95	0.99	0.97	1297
1	0.91	0.68	0.78	203
avg / total	0.95	0.95	0.94	1500

Figure 3.2: Model Results 1/4

code here:

The results shows a good accuracy however the recall for these models are not acceptable as we are more focused on the churning ratio and if the models predicts recall of 74% for Decision Tree, 27% for K-Nearest Neighbors, 18% for Logistic Regression and 68% for Random Forest Model. Which are relatively less and leads to the Type I Error which means the model gives irrelavant measure in predicting that customers' churn however they don't i.e; a minimum accuracy in predicting the False Positive Rate. The given ROC depicts the Area Under Curve for different models. All models didn't perform well and hence they fail to give a better AUC.

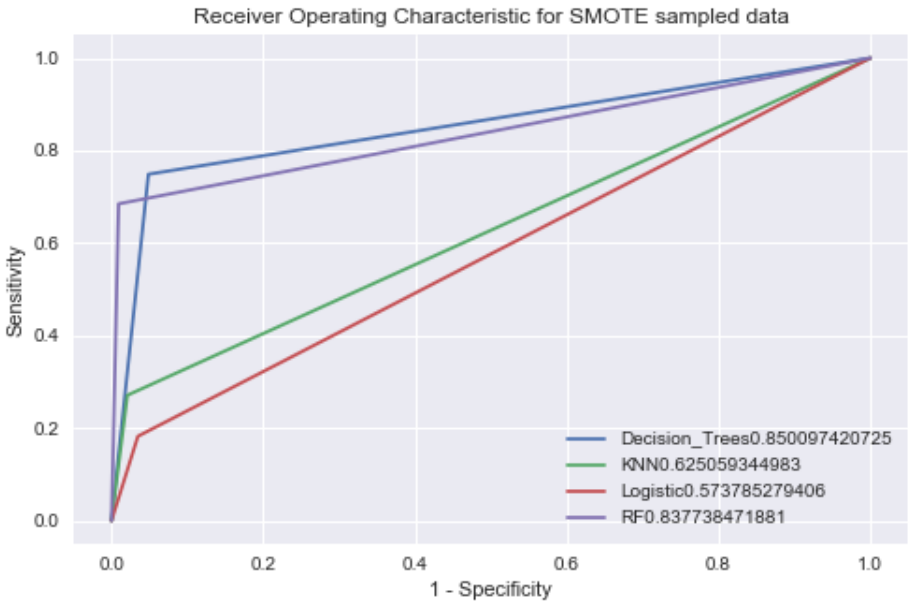

Figure 3.3: Initial ROC plot(See Python Code Fig: B.1)

Upon Re-Analysing we see that, the dataset is more biased towards the Customers who didn't Churned out then who did. Which leads to Target Class Imbalance in the dataset.

3.2 Target Class Imbalance

It is the problem in machine learning where the total number of a class of data (positive) is far less than the total number of another class of data (negative). This problem is extremely common in practice and can be observed in various disciplines.

Most machine learning algorithms and works best when the number of instances of each classes are roughly equal. When the number of instances of one class far exceeds the other, problems arise. This is best illustrated with our current situation.
In our dataset of Churn ratio, we would like to find out which are Customers that churned and who didn't. Now, it is highly cost effective for telecom company if a trusted customers churns, and costs a loss of a valuable customer. We want to catch as many cases of TRUE churning as possible and then optimize the parameters to reduce the TRUE churning ratio.
The imbalanced ratio as shown

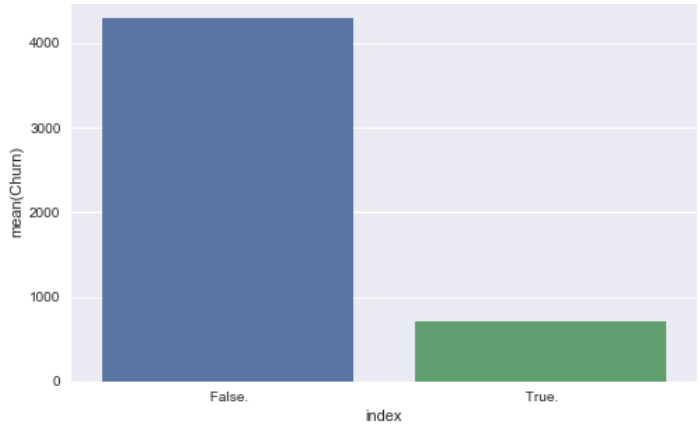

Figure 3.4: Target Class Imbalance

3.3 SMOTE for Reducing Imbalance

Synthetic Minority Oversampling (SMOTE) is a technique which resamples the minority samples by selecting k nearest minority samples and then creating a synthetic sample which equals to the average between two samples, it's better option then replicating minority samples.
We will apply this sampling approach and check every models' performance. The R and Python implementations are as shown

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 101)
X_res, y_res = sm.fit_sample(X, y)
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res
    , test_size=0.3, random_state=101)
```

Listing 3.1: Python SMOTE Implementation

```r
#Creating over,under,both and synthetic samples to overcome
    target imbalance
cdf_over = ovun.sample(Churn ~., data = cdf_train, method = '
    over',N = 5984)$data
cdf_under = ovun.sample(Churn ~., data = cdf_train, method = '
    under',N = 1004)$data
cdf_both = ovun.sample(Churn ~., data = cdf_train, method = '
    both',
                       p = 0.5,
                       seed = 221,
                       N = 3494)$data

cdf_ROSE = ROSE(Churn ~., data = cdf_train,
                N = 5000,
                seed = 221)$data
```

Listing 3.2: R resampling

After Resampling the data we see that there is an equal distribution for both the categories in the response variable.

```
In [84]: dv2 = pd.DataFrame(y_res)
         dv2.rename(index = str, columns= {0:'res'},inplace = True)
         dv2['res'].value_counts()

Out[84]: 1    4293
         0    4293
         Name: res, dtype: int64
```

Figure 3.5: Balanced resampled data

3.4 Re-modeling with Balanced Set

After resampling using the Synthetic Method the models are trained with the a balanced target class and hence the obatined ROC is as shown, which shows better AUC and the models perform well in determining the Recall and the Precision alongwith a better Accuracy. The RF model again shows a better accuracy of 97% while Logistic model with an acuuracy of 79% being the least. The Receiver Operator Curve for these models is as shown

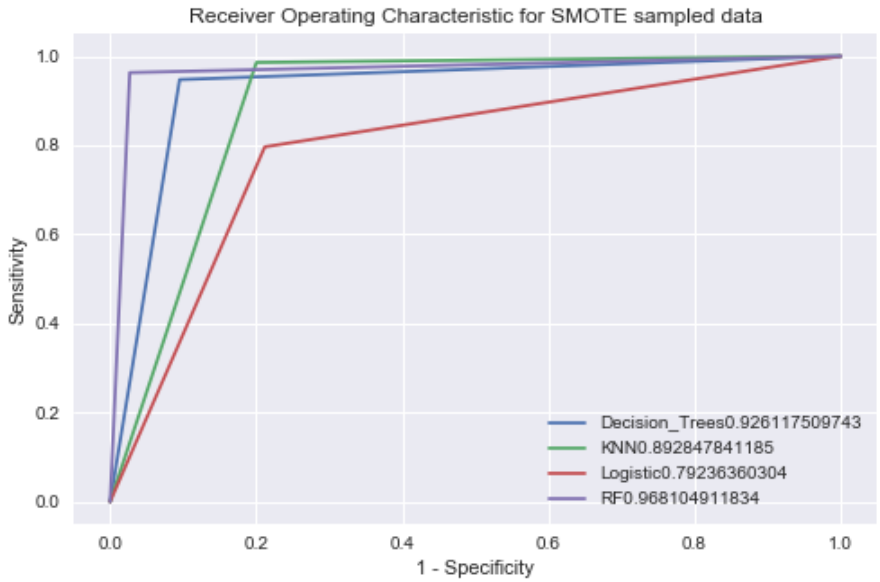

Figure 3.6: ROC on Balanced Data

The shown ROC curve depicts better performance with a balanced dataset. There is significant difference in both the ROC curve and we can conclude that Target Class Imbalance has made the model to perform with a better Recall and hence the aim of Reducing the Churn with a decent Recall solves the issue.

Chapter 4

Conclusion

4.1 Model Evaluation

The metrics to evaluate a machine learning model is very important. Choice of metrics influences how the performance of machine learning algorithms is measured and compared. We can use classification performance metrics such as:
Log-Loss
Accuracy
AUC(Area under Curve) etc
Another example of metric for evaluation of machine learning algorithms is precision, recall, which can be used for sorting algorithms primarily used by search engines.

4.1.1 Precision

With the evaluation on all the models we can conclude that our predictive performance parameter is the Precision Rate by which we can accurately predict the number of customers that will churn out.

```
> confusionMatrix(ConfMatrix_RF)
Confusion Matrix and Statistics

  RF_Predictions
      0    1
  0 1253    9
  1   40  173

               Accuracy : 0.9668
                 95\% CI : (0.9563, 0.9753)
```

```
    No Information Rate : 0.8766
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8569
 Mcnemar's Test P-Value : 1.822e-05

            Sensitivity : 0.9691
            Specificity : 0.9505
         Pos Pred Value : 0.9929
         Neg Pred Value : 0.8122
             Prevalence : 0.8766
         Detection Rate : 0.8495
   Detection Prevalence : 0.8556
      Balanced Accuracy : 0.9598

       'Positive' Class : 0
```

Listing 4.1: RF performance

Better predictions on the Churning ratio is obtained with the Random Forest Model and hence it can be selected as the final model to precisely predict upon the parameters to that can lead to the cancellation of service by a customer.

4.1.2 AUC/ROC

The final model that can be perfectly classify among the two kinds of customer is the Random Forest Model.

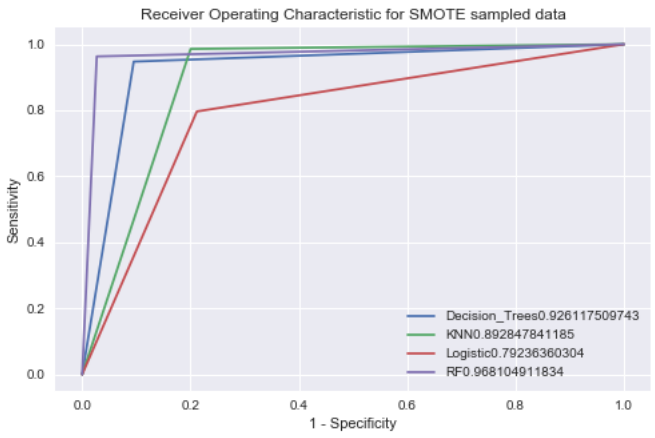

Figure 4.1: ROC on Balanced Data

4.2 Model Selection

With the obtained results we can say the model on Random FOrest is a decent predictor of the Churn Class and we can finalize this model for the approach ROC Fig: A.4 .

4.2.1 Important Variables

The important variables for the churn are:
1. Total day Minutes which relates to the day charges
2. International Plan
3. Number of Customer Service Calls

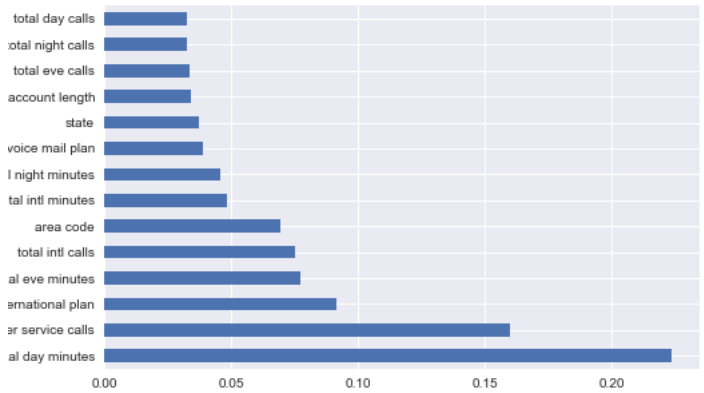

Figure 4.2: Important Variables in Analysis

Python Code here: B.7

Appendix A

Extra Figures

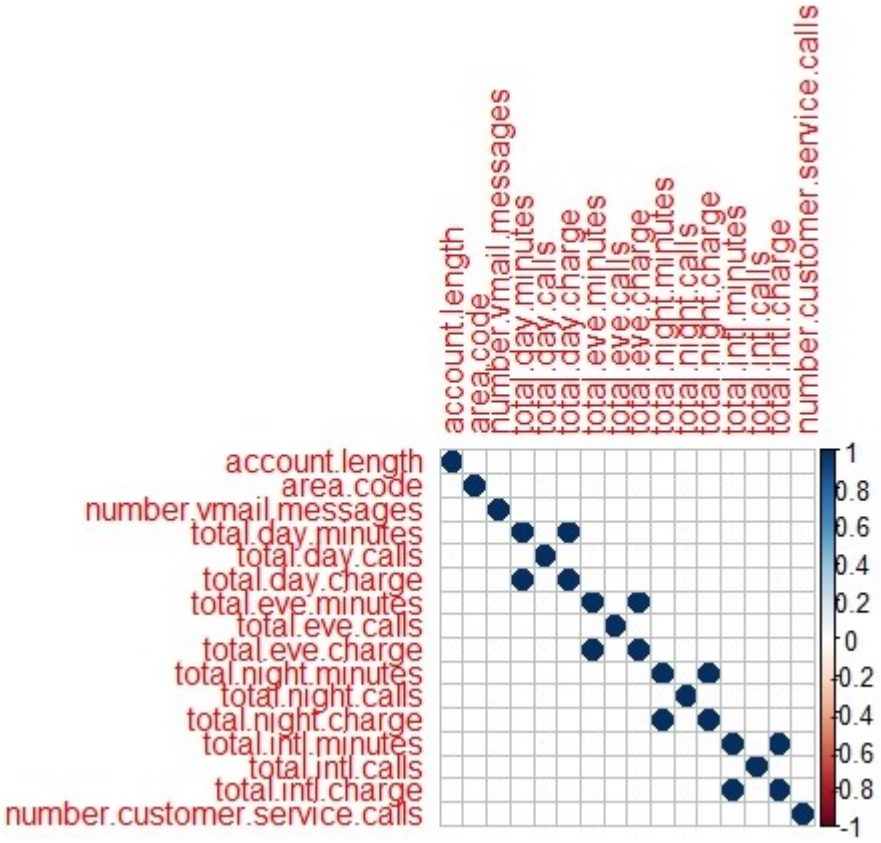

Figure A.1: R corrplot (See R code in appendix B.5)

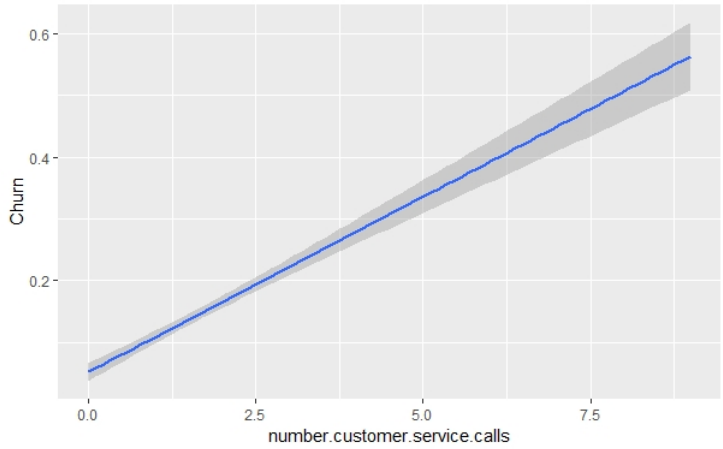

Figure A.2: Linear Graph for relation with target variable (See R code in appendix B.4)

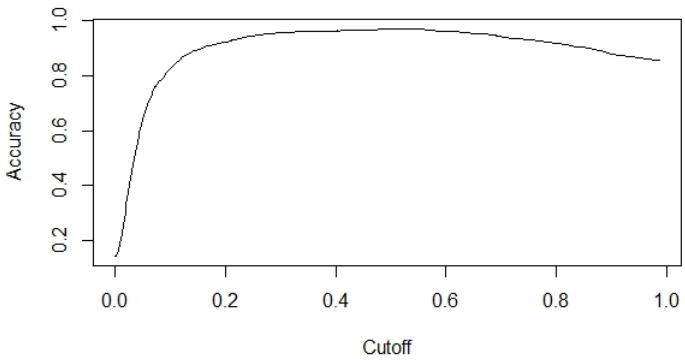

Figure A.3: ROC for the Random Forest Model (See R code in appendix B.2)

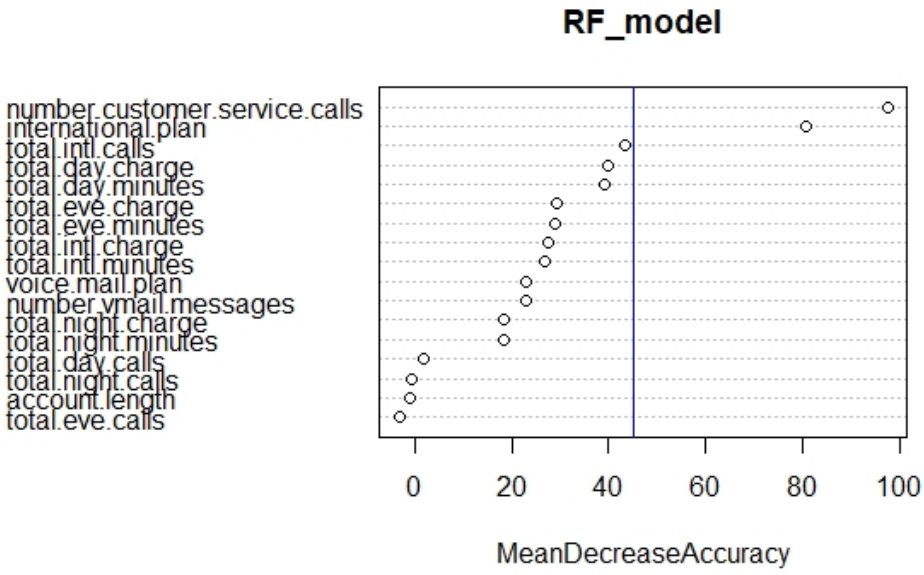

Figure A.4: Variable Importance for Random Forest(See R code in appendix B.3)

Appendix B

Code

```

model_predictions = [pred1,pred2,pred3,pred4] #list for every
    models predictions
models = ['Decision_Trees','KNN','Logistic','RF'] #list to
    specify the model names

#looping multiple models for plotting a single ROC Curve to
    evaluate models performance
for plots in range(0,4):
    fpr, tpr, thresh = metrics.roc_curve(y_test,
    model_predictions[plots])
    auc = metrics.roc_auc_score(y_test, model_predictions[plots
    ])
    plt.plot(fpr,tpr,label=models[plots]+str(auc))
    plt.legend(loc=0)
    plt.xlabel('1 - Specificity')
    plt.ylabel('Sensitivity')
    plt.title('Receiver Operating Characteristic for SMOTE
    sampled data')
    print(models[plots])
    print(classification_report(y_test,model_predictions[plots])
    )
    plt.savefig('Balanced_ROC.png')
```

Listing B.1: Multiple ROC Model

```

#ROC Curve
RF_roc = predict(RF_model,cdf_test,type = 'prob')[,2]
RF_roc =  prediction(RF_roc,cdf_test$Churn)
eval_ = performance(RF_roc,'acc')
plot(eval_)
```

Listing B.2: RF ROC

```
#Variable Importance
plot.new()
varImpPlot(RF_model,type = 1)
abline(v = 45, col= 'blue')
#This plot resembles the important parameters in RF prediction
```

Listing B.3: RF ROC

```
# Correlation with the Target Variable
ggplot(customer_df, aes(x=international.plan, y=Churn)) +
  geom_point(shape=1) +
  geom_smooth(method=lm)

ggplot(customer_df, aes(x=total.day.minutes, y=Churn)) +
  geom_smooth(method=lm)

ggplot(customer_df, aes(x=total.day.charge, y=Churn)) +
  geom_smooth(method=lm)

ggplot(customer_df, aes(x=number.customer.service.calls, y=Churn
    )) +
  geom_smooth(method=lm)
#All these plots shows a positive relation with the Target
    Variable
```

Listing B.4: Relation Plot

```
corrgram(customer_df, order = F, lower.panel = panel.shade,
         upper.panel=panel.pie, text.panel=panel.txt, main = "
    Correlation Plot")
```

Listing B.5: Correlation Plot

```
states = states_.value_counts()
state_ls = list(states.index)
state_vs = list(states.values)
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode
    , plot, iplot
init_notebook_mode(connected=True)
data = dict(type='choropleth',
            colorscale = 'Picnic',
            locations = state_ls,
            z = state_vs,
            locationmode = 'USA-states',
            text = target,
            marker = dict(line = dict(color = 'rgb(255,255,255)'
    ,width = 2)),
            colorbar = {'title':"Total Customers"}
```

```
            )
layout = dict(title = 'US Customers Heatmap by State',
              geo = dict(scope='usa'))
choromap = go.Figure(data = [data],layout = layout)
iplot(choromap)
```

Listing B.6: US Heatmap

```
vars_ = (pd.Series(model_RF.feature_importances_, index=X.
    columns)
   .nlargest(14)
   .plot(kind='barh'))
plt.savefig('varz_imp.png')
```

Listing B.7: Variable Importance